PR #35175 完整报告

sgl-project/sglang

[DSA] Route the ragged prefill top-k to the v2 kernel

合并时间：2026-08-22 07:59

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35175

执行摘要

- 一句话：RAGGED prefill top-k 切换至 v2 JIT 内核，extend 提速 1.3-1.8×
- 推荐动作：值得精读。该 PR 是 CUDA kernel 性能优化的优秀样本：bias 字段配合编译期折叠实现零开销接口扩展；in-place 掩码把未对齐窗口的代价压到最低；测试用哨兵分数让任何窗口外泄漏立即暴露。建议重点阅读 topk_v2.cuh 中 topk_ragged_kernel 的注释论证、dsa_topk_backend.py 的 dispatch 条件，以及 test_topk_v2.py 的 OUTSIDE_SCORE 设计与写范围断言。合并前需确认 AMD ROCm CI 失败是否为既有 known failure。

功能与动机

PR body 明确指出：The extend-shaped RAGGED transform is the last DSA top-k path still served by the legacy sgl-kernel kernel (topk_transform_prefill_ragged_kernel), even though it is the cheapest one to serve. DeepGEMM contiguous-KV 索引器输出的列已经是 batch 扁平化 KV 中的绝对位置，RAGGED 变换不需要 page table，本质上只是每行一次加法，正好是 v2 kernel raw-index 模式（#35041 引入）加上一个 bias 就能覆盖的场景。因此作者将这条最后的 legacy 路径路由到 v2，既消除最后一段 legacy 依赖，又换取约 1.3-1.8× 的 prefill 加速。

实现拆解

1. 内核参数扩展：bias 字段（python/sglang/kernels/jit/include/sgl_kernel/deepseek_v4/topk_impl.cuh）。TopKProblem 新增默认值为 0 的 int32_t bias，emit() 对每个选中索引统一加 bias。既有所有构造点都省略该字段，编译器可 constant-fold 掉加法；PR 用 cuobjdump -sass 对比前后各 12 个 kernel / 56648 条指令，证明 paged 路径 11 个 kernel 字节级一致，仅 cluster kernel 出现 +1 IMAD.MOV.U32 / -1 NOP 的指令置换，资源占用完全一致。
2. 新增 ragged kernel（python/sglang/kernels/jit/csrc/deepseek_v4/topk_v2.cuh）。新增 TopKRaggedParams 与 topk_ragged_kernel：每个 block 处理一行，在 [row_starts[b], row_starts[b] + seq_lens[b]) 窗口内选 top-k，输出 position + out_offsets[b]。seq_len <= topk 走 trivial 路径直接写出、不读 scores；未对齐窗口把读基址向下取整到 4-float 边界，并将提前读入的 ≤ 3 列掩码为 -FLT_MAX，掩码落在 PDL wait 之后、由 forward() 开头的 __syncthreads() 发布；不引入 cluster 路径（prefill 行数多，单行多 block 无收益）。宿主侧新增 TopKKernel::transform_ragged。

3. Python JIT 入口（python/sglang/kernels/ops/attention/dsv4/topk.py）。新增 topk_transform_ragged_v2 包装函数，JIT wrapper 注册 topk_transform_ragged；原 topk_transform 改名 topk_transform_paged，topk_transform_512_v2 同步改调新名称。docstring 明确 scores 会被原地写、seq_lens 必须非负。
4. 后端路由（python/sglang/srt/layers/attention/dsa/dsa_topk_backend.py + dsa_indexer_metadata.py）。DSATopKBackend.topk_transform 的 RAGGED 分支新增 dispatch 条件：should_use_topk_v2()、topk_indices_offset 非空、batch_idx_list is None、0 < topk <= 2048、三张表行数一致；满足时调用新增的 _topk_transform_v2_ragged。batch_idx_list 非 None 的 prefill-CP 路径必须留在 legacy kernel，因为其 topk_indices_offset 由 cu_seqlens_q 构造而非 KV base。同时 BaseIndexerMetadata.topk_transform 增加 **kwargs 透传 row_starts 等可选参数。
5. 测试与基准（test/registered/kernels/ops/attention/test_topk_v2.py、test/registered/kernels/benchmark/attention/bench_topk.py）。新增 44 个 ragged 用例：覆盖 row_start % 4 全残差 × 全部模板带（trivial / Register2 / Register4 / Streaming）、一次 launch 的混合长度、选择边界（seq == k、k + 1、8192/8193、16385）、长上下文 131072、row_starts=None，以及一个与 row_starts 不同的 offset；窗口外填充 OUTSIDE_SCORE = 1e3 哨兵值，任何泄漏都会表现为错误选择，并额外断言只有掩码头部被合法写入。benchmark 侧新增 benchmark_ragged，同时修复重复函数名导致 benchmark_paged 被 body-less stub 遮蔽的缺陷。

关键文件：

- python/sglang/kernels/jit/csrc/deepseek_v4/topk_v2.cuh（模块 JIT 内核；类别 source；类型 core-logic；符号 TopKRaggedParams, topk_ragged_kernel, TopKKernel::transform_ragged, TopKKernel::transform_paged）：新增 TopKRaggedParams、topk_ragged_kernel 与宿主侧 transform_ragged，是本次变更的核心 kernel 实现；未对齐窗口掩码、PDL 时序、trivial 路径与无 cluster 路径的设计都在这里。
- python/sglang/srt/layers/attention/dsa/dsa_topk_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 _topk_transform_v2_ragged, topk_transform）：RAGGED 分支路由的核心：新增 dispatch 条件与 _topk_transform_v2_ragged 包装，决定生产路径何时切换到 v2 kernel、何时必须留在 legacy。
- python/sglang/kernels/jit/include/sgl_kernel/deepseek_v4/topk_impl.cuh（模块 JIT 内核；类别 source；类型 core-logic；符号 TopKProblem::emit, TopKProblem::bias）：TopKProblem 新增 bias 字段是 ragged 模式输出变换的基础；默认 0 使既有构造点在编译期折叠加法，是保持既有 paged kernel SASS 字节级不变的关键设计。
- test/registered/kernels/ops/attention/test_topk_v2.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_ragged, _run_ragged, test_topk_v2_ragged_window, test_topk_v2_ragged_no_row_starts）：新增 44 个 ragged 用例，覆盖所有对齐残差、模板边界、混合长度、长上下文与 row_starts=None，并用哨兵分数与写范围断言保证任何窗口外泄漏都会失败。
- python/sglang/kernels/ops/attention/dsv4/topk.py（模块 算子接口；类别 source；类型 entrypoint；符号 topk_transform_ragged_v2, topk_transform_paged,

_jit_topk_v2_module）：Python 侧 JIT 入口：新增 topk_transform_ragged_v2，注册 topk_transform_ragged wrapper，并把原 topk_transform 改名 topk_transform_paged，是后端路由与 kernel 之间的接口层。

- test/registered/kernels/benchmark/attention/bench_topk.py（模块 性能基准；类别 test；类型 test-coverage；符号 benchmark_paged, benchmark_ragged, _build_paged_fn, _build_ragged_fn）：新增 benchmark_ragged 与 _build_ragged_fn，让 ragged 路径的 jit_v1 / jit_v2 / flashinfer 可对照；同时修复了重复函数名导致 benchmark_paged 被 body-less stub 遮蔽的缺陷。
- python/sglang/srt/layers/attention/dsa/dsa_indexer_metadata.py（模块 索引器元数据；类别 source；类型 data-contract；符号 BaseIndexerMetadata.topk_transform）：BaseIndexerMetadata.topk_transform 增加 **kwargs 透传，使后端可以传递 row_starts 等可选参数，是接口层面的配套调整。

关键符号：TopKProblem::emit, TopKProblem::bias, topk_ragged_kernel, TopKKernel::transform_ragged, TopKKernel::transform_paged, topk_transform_ragged_v2, topk_transform_paged, _topk_transform_v2_ragged, DSATopKBackend.topk_transform, BaseIndexerMetadata.topk_transform, test_topk_v2_ragged_window, test_topk_v2_ragged_no_row_starts, benchmark_ragged, benchmark_paged

关键源码片段

python/sglang/kernels/jit/csrc/deepseek_v4/topk_v2.cuh

新增 TopKRaggedParams、topk_ragged_kernel 与宿主侧 transform_ragged，是本次变更的核心 kernel 实现；未对齐窗口掩码、PDL 时序、trivial 路径与无 cluster 路径的设计都在这里。

```
// topk_ragged_kernel：ragged（prefill）top-k，每行一个 block。
// 行 b 在 [row_starts[b], row_starts[b] + seq_lens[b]) 内选 top-k，
// 输出 = 选中位置 + out_offsets[b]，其余位置填 -1。
template <bool kPDL>
TOPK_KERNEL void topk_ragged_kernel(const __grid_constant__ TopKRaggedParams params) {
    device::enable_smem_spilling();
    constexpr uint32_t kVecSize = impl::TopKStreaming::kVecSize; // 16 字节向量宽度 = 4 个 float
    const auto bx = blockIdx.x;

    // 提前发起元数据预取
    const auto seq_len = static_cast<uint32_t>(params.seq_lens[bx]);
    const auto offset = params.out_offsets[bx];
    const auto row_start = params.row_starts == nullptr ? 0u : params.row_starts[bx];
    const auto topk = params.topk;
    const auto out = params.topk_indices + bx * static_cast<int64_t>(topk);

    // trivial 路径：行内 token 数不超过 topk，直接顺序写出，
    // 不读 scores，因此完全不涉及对齐与掩码。
    if (seq_len <= topk) {
        device::PDLWaitPrimary<kPDL>();
```

```
        for_each_item(topk, [&](uint32_t tx, uint32_t) {
            out[tx] = tx < seq_len ? static_cast<int32_t>(tx) + offset : -1;
        });
        return;
    }

    // 未对齐窗口：row_start 是任意 token offset，向量化读需要把基址
    // 向下取整到 4-float 边界，提前拉入的 <= 3 列属于同一行的前一个
    // 请求，是真实有限分数，必须在本行选择前掩成 -FLT_MAX。
    const auto rem = row_start % kVecSize;
    const auto score = params.scores + bx * params.score_stride;
    if (rem != 0) {
        // 掩码必须落在 PDL wait 之后，否则会被上游 DeepGEMM 索引器覆盖；
        // 它由后续 forward() 开头的 __syncthreads() 发布，无需额外 barrier。
        device::PDLWaitPrimary<kPDL>();
        static_assert(kVecSize <= kBlockSize, "not enough threads");
        if (const auto tx = threadIdx.x; tx < rem) {
            score[row_start - rem + tx] = -std::numeric_limits<float>::max();
        }
    }

    // 把窗口整体平移 rem：读取基址对齐，输出 bias 同步减去 rem，
    // 保证"选中位置 + offset"语义不变。
    const auto problem = TopKProblem{
        .in = score + (row_start - rem),
        .out = out,
        .page_table = nullptr, // 未使用
        .topk = topk,
        .seq_len = seq_len + rem,
        .page_bits = 1, // 未使用
        .bias = offset - static_cast<int32_t>(rem),
    };
    __shared__ impl::MaxSmem<Register2::Smem, Register4::Smem, Streaming::Smem> smem;
    if (problem.seq_len <= Register2::kMaxSeqLen) {
        Register2::forward<kPDL>(problem, &smem);
    } else if (problem.seq_len <= Register4::kMaxSeqLen) {
        Register4::forward<kPDL>(problem, &smem);
    } else {
        Streaming::forward<kPDL>(problem, &smem);
    }
    // 本 kernel 只处理单行，PDL secondary trigger 没有用途，忽略
}
```

python/sglang/srt/layers/attention/dsa/dsa_topk_backend.py

RAGGED 分支路由的核心：新增 dispatch 条件与 _topk_transform_v2_ragged 包装，决定生产路径何时切换到 v2 kernel、何时必须留在 legacy。

```
# RAGGED 分支路由：仅当满足全部条件时才切换到 v2 ragged kernel。
# 核心约束是 batch_idx_list 必须为 None——prefill-CP 路径的
```

```
    # topk_indices_offset 基于 cu_seqlens_q 而非 KV base，语义不同，
    # 必须继续走 legacy kernel。
    if (
        self.should_use_topk_v2()
        and topk_transform_method == TopkTransformMethod.RAGGED
        and topk_indices_offset is not None
        and batch_idx_list is None
        and 0 < topk <= 2048
        and lengths.shape[0] == logits.shape[0] == topk_indices_offset.shape[0]
    ):
        return _topk_transform_v2_ragged(
            logits, lengths, topk, topk_indices_offset, row_starts
        )


    def _topk_transform_v2_ragged(
        logits: torch.Tensor,
        lengths: torch.Tensor,
        topk: int,
        topk_indices_offset: torch.Tensor,
        row_starts: Optional[torch.Tensor],
    ) -> torch.Tensor:
        """通过 DeepSeek-V4 v2 JIT kernel 完成融合 ragged top-k。

        logits 会被原地写：kernel 从 16 字节对齐基址读取，并把窗口前
        多读的 <= 3 列掩码为极小值；这些列属于同一行前一个请求，
        且该分数缓冲在 top-k 之后即被丢弃
        （参见 DSAIndexer._get_topk_ragged）。
        前置条件与 paged 入口一致：fp32、unit row stride、16B 对齐 stride。
        """
        from sglang.kernels.ops.attention.dsv4.topk import topk_transform_ragged_v2

        out = logits.new_empty((logits.shape[0], topk), dtype=torch.int32)
        topk_transform_ragged_v2(
            logits,
            lengths,
            out_offsets=topk_indices_offset,
            out_indices=out,
            row_starts=row_starts,
        )
        return out
```

python/sglang/kernels/jit/include/sgl_kernel/deepseek_v4/topk_impl.cuh

TopKProblem 新增 bias 字段是 ragged 模式输出变换的基础；默认 0 使既有构造点在编译期折叠加法，是保持既有 paged kernel SASS 字节级不变的关键设计。

```
// TopKProblem 是 v2 内核各模板（Register2 / Register4 / Streaming / Cluster）
// 共享的问题描述结构。新增的 bias 字段专门服务 ragged 模式：
// 输出 = 选中位置 + bias。
```

```
struct TopKProblem {
    uint32_t topk;
    uint32_t seq_len;
    uint32_t page_bits;
    int32_t bias = 0; // ragged 模式专用输出偏移；paged 模式始终保持 0

    // 所有既有构造点都省略 bias，因此编译器能把这个加法 constant-fold 掉；
    // 这保证 paged 路径的 SASS 与改动前字节级一致（PR 中已用 cuobjdump 验证）。
    SGL_DEVICE void emit(uint32_t pos, uint32_t raw_idx) const {
        out[pos] = static_cast<int32_t>(raw_idx) + bias;
    }

    SGL_DEVICE void transform_output(uint32_t t, int32_t raw) const {
        // paged 模式：raw < 0 时填 -1，否则经 page table 换算为 KV 位置
        out[t] = raw < 0 ? -1 : page_to_indices(page_table, raw, page_bits);
    }
};
```

评论区精华

本 PR 没有任何 review 评论（comments_count=0、review_comments_count=0），以下提炼自 PR body 与 commit message 中作者给出的技术论证：

- 关于未对齐窗口掩码的并发安全，作者论证：The row belongs to this block alone and the score buffer is dead after the top-k, so the write races with nothing. It lands after the PDL wait (the indexer rounds its own KV base down to 4 as well and would otherwise overwrite the mask) and is published by the __syncthreads() that every forward() already runs before it reads any score.
- 关于 trivial 路径：It cannot reuse trivial_transform, whose -1 padding would pick up the bias.
- 关于 prefill-CP 保留 legacy：its topk_indices_offset is built from cu_seqlens_q rather than the KV bases，语义不同，不能路由到 v2。
- 关于性能取舍：1.3–1.8× over the legacy kernel, except the single-row 64K shape where the cluster split flashinfer/the paged path use still wins — not a prefill shape。
- 关于 SASS 回归验证：12 个 kernel / 56648 条指令对比，11 个 byte-identical，仅 cluster kernel 有一条指令置换，cuobjdump -res-usage 字段级完全一致。
- 未对齐窗口掩码的原地写安全性 (correctness): 作者论证：一行只属于一个 block、分数缓冲在 top-k 之后即被丢弃、掩码落在 PDL wait 之后且由 forward() 开头的 __syncthreads() 发布；测试通过写范围断言进一步固定该约定。
- bias 字段对既有 kernel 的零开销扩展 (design): 默认 0 让编译期 constant-fold；cuobjdump 对比实现 12 个 kernel 中 11 个字节级一致、1 个仅指令置换（IMAD.MOV.U32 换 NOP），资源占用完全一致。
- ragged 模式不做 cluster 路径 (design): prefill 行数成千上万，而 cluster 只对极少数行的 decode 形状有价值，因此新 kernel 采用每行一个 block，省去矩阵与同步开销。

- prefill-CP 分支为何保留 legacy (correctness): 其 topk_indices_offset 由 cu_seqlens_q 构造而非 KV base，语义不同；强行路由会产生错误索引，故限制 batch_idx_list 为 None 才走 v2。
- 单行 64K 形状反而更慢 (performance): 该形状是 cluster split 的典型受益者，但 prefill 的行数是 extend token 数，单行长行不属于 prefill 实际工作负载，可接受。

风险与影响

- 风险：核心 prefill 路径切换：RAGGED 是 DSA extend 请求的必经路径，路由条件一旦误判（例如未来出现新的 offset 语义），会把错误形状送到 v2 kernel；当前通过 batch_idx_list is None 与行数一致双重防护，但后续改动需保持警惕。in-place 写分数缓冲：topk_ragged_kernel 会原地掩码窗口前的 ≤ 3 列，依赖"分数缓冲在 top-k 后即死"的调用约定（DSAIndexer._get_topk_ragged），若未来索引器复用缓冲将产生跨请求污染。时序耦合：掩码必须在 PDL wait 之后落盘，否则会被上游 DeepGEMM 覆盖，属于隐蔽时序约束。双路径共存：prefill-CP 仍走 legacy kernel，边界需长期维护。CI 状态：AMD ROCm 7.2 运行失败（Run #32349896357），需确认是否属于 KNOWN_FAILURES.md 既有失败。兼容性：bench_topk.py 中 benchmark 改名 benchmark_paged，外部引用旧函数名的脚本会失效。
- 影响：用户侧：DeepSeek-V4（DSA 索引器）extend 请求的 top-k 延迟在 B200 k=2048 下降低约 1.3-1.8×，且 topk_indices_offset == row_starts 时输出即 token 在扁平化 KV 中的绝对位置，语义直观。系统侧：DSA top-k 的 decode PAGED 与 extend RAGGED 两条路径均已切换到 v2，进一步减少对 legacy sgl-kernel 的依赖。团队侧：kernel 维护重心向 v2 收敛，但引入两条路径并存期与 in-place 写约定，评审和后续重构需要更高警惕。测试资产增强明显：44 个新用例覆盖全部对齐残差与模板边界，为后续 kernel 演进提供回归基线。
- 风险标记：核心 prefill 路径切换，in-place 写分数缓冲，未对齐掩码依赖 PDL 时序，AMD ROCm CI 待确认，prefill-CP 双路径共存

关联脉络

- PR #35041 [DSA] Add raw-index mode to the v2 top-k kernel: PR body 明确声明本 PR stacked on #35041，且 v2 的 raw-index 模式由它引入；本 PR 在其基础上补充 bias 后即可覆盖 ragged 场景。注意该 PR 不在本次历史列表中，标题为据 body 推断。